Extended Cognition Hypothesis Applied to Computational Thinking in Computer Science Education

Letonsaari, Mika

Springer
2018-06-12

Letonsaari , M 2018 , Extended Cognition Hypothesis Applied to Computational Thinking in Computer Science Education . in Y Shi , H Fu , Y Tian , V V Krzhizhanovskaya , M H Lees , þÿ J Dongarra & P M A Sloot (eds) , Computational Science ICCS 2018 : Conference Wuxi, China, June 11-13, 2018, Proceedings, Part III . vol. Part 3 , Lecture Notes in Computer Science , no. 10862 , Springer , Cham , pp. 304-317 , The International Conference on Computational Science (ICCS 2018) , Wuxi , China , 11/06/2018 . https://doi.org/10.1007/978-3-319-9

http://hdl.handle.net/10138/321425
https://doi.org/10.1007/978-3-319-93713-7_25

unspecified
acceptedVersion

Extended Cognition Hypothesis Applied to Computational Thinking in Computer Science Education

Mika Letonsaari[1,2]

[1] University of Helsinki, Helsinki, Finland mika.letonsaari@helsinki.fi
[2] South-Eastern Finland University of Applied Sciences, Mikkeli, Finland

Abstract. Computational thinking is a much-used concept in computer science education. Here we examine the concept from the viewpoint of the extended cognition hypothesis. The analysis reveals that the extent of the concept is limited by its strong historical roots in computer science and software engineering. According to the extended cognition hypothesis, there is no meaningful distinction between human cognitive functions and the technology. This standpoint promotes a broader interpretation of the human-technology interaction. Human cognitive processes spontaneously adapt available technology enhanced skills when technology is used in cognitively relevant levels and modalities. A new concept technology synchronized thinking is presented to denote this conclusion. More diverse and practical approach is suggested for the computer science education.

Keywords: computational thinking, extended cognition, externalism, philosophy of mind, computer science education

1 Introduction

In this article, we study the concept of computational thinking from the viewpoint of the extended cognition hypothesis. Both of these concepts have deep roots in the history of the 20th-century science but they have only very recently made a major impact in the mainstream science.

Computational thinking is a broad term describing the skills needed for solving problems with computers and computational methods. The concept has been popular for about a decade. It has been widely adopted in computer science education and generally made into one of the key concepts in 21st-century skills in education.

Externalism is a school of thought in philosophy. The main idea of externalism is that consciousness and cognition cannot be understood only as the result of the function of the brain or nervous system. The extended cognition hypothesis is an active form of externalism claiming that things external to the human body function as parts of the human cognitive function. The idea has been widely discussed especially during last two decades, and it has had a prominent contribution to the philosophy of mind, cognitive science, and psychology.

An extensive review of these concepts is out of the scope of this article. Both concepts are introduced here briefly from a practical view of computer science education.

1.1 Computational Thinking

The history of computational thinking is related to the birth of computer science in the early 20th century. Before computer science was accepted as a separate discipline it operated under the disciplines of electric engineering and mathematics. But as the importance of computers and programming grew, there was a need to define what separates computer science from other disciplines.

The distinction between disciplines is often emphasized by the use of different mental processes in the way the disciplines are practiced. This novel way of thinking was called "algorithmizing" by Alan Perlis and it was described as a "general purpose thinking tool" by George Forsythe in the 1960s. In the 1970s Donald Knuth used a term "algorithmic thinking" [1].

For example, E. W. Dijkstra discussed the problems of using mathematical proofs in understanding programming in 1974 and observed three differences in the required thinking: construction of new concepts, construction of new notations, and level of abstraction in the terms of semantics [2].

Donald Knuth concluded that the algorithmic thinking is the main difference between mathematics and computer science [3] and later specified practical aspects of algorithmic complexity as the key difference [4]. In the same article Knuth pointed out the use of assignment operator as a distinctive example. The assignment operator in computer programming is often the equals sign of mathematics and equality has often typographically more complex symbol [1].

The term computational thinking was first used by Seymour Papert in 1980 [5]. It became widely popular in 2006 when Jeannette Wing published an essay where she suggested that computational thinking was a beneficial skill set for everyone, not only computer scientists [6].

There are many definitions of the term computational thinking. Wing uses the following definition developed with Jan Cuny, Larry Snyder, and Alfred Aho: "The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." [7, 8]

A more specific definition, breaking the term into its constituents in the practical context of education, is developed by the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE). According to their definition computational thinking includes the following elements [9]:

- formulating problems in a way that enables us to use a computer and other tools to help solve them

[1] While there is a traditionally used markup for the assignment, ':=', the equality sign '=' is often used as an assignment operator in computer languages. Equality is tested with double equality sign '==' or functions like 'equals()' or 'is_equal()'.

- logically organizing and analyzing data
- representing data through abstractions such as models and simulations
- automating solutions through algorithmic thinking (a series of ordered steps)
- identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- generalizing and transferring this problem-solving process to a wide variety of problems

There are other definitions and critique of the definition, especially for the vagueness and interpretations of the definition. For a comprehensive review of the subject see Tedre [10].

While no final definition has been achieved for the term, there is a wide consensus that the term computational thinking encapsulates something essential in the deeper understanding of the computer science and the modern technological world. And for this reason, there has been much work to incorporate the ideas of computational thinking into k-9 and k-12 computer science education.

1.2 Extended Cognition Hypothesis

The relationship between human mind and human body has always fascinated artists, scholars of religions, and scientists. Religious scriptures include some of the oldest references to the subject. Parmenides of Elea in Greece in the 5th century BCE, and later Plato, brought the issue to the systematic philosophical discussion [11]. This was followed by probably the most well-known philosophical debate in Western philosophy, the debate between Cartesian dualism, especially defended by René Descartes, and different schools of monism [12].

It was not until late 20th century that technological development in biology, medical imaging, and cognitive science managed to reveal the function of the brain and how cognitive functions arise from neural activity [13]. By the end of the 20th century, there was a significant demand for an explanation of the relationship between human mind and its environment [14]. This demand culminated in the publication of *The Extended Mind* paper in 1998 by David Chalmers and Andy Clark where the extended mind hypothesis was presented [15].

Extended cognition hypothesis is a philosophical position that cognitive processes and human mind extend beyond the brain in the environment. As a simple example, one can write things down with a pen and a paper so that one doesn't need to remember them. Or one can use a pocket calculator to perform mathematical calculations.

Using external tools human cognition saves the use of memory and processing capacity. Cognitive processing is extended outside the human body. Extended cognition hypothesis claims that there is no meaningful distinction between internal and external processing for cognitive processes.

According to Mark Rowlands, mental processes of the extended mind can be further classified as embodied, embedded, enacted, and extended processes [16].

The classification describes the relationship of the human neural system, the human body, and the environment, as well as the possible actions of the entities.

These ideas have had a major impact in several disciplines such as cognitive science [17], psychology [18], and biology [19]. In the educational sciences extended cognition hypothesis and especially the concept of enactivism is closely related to constructivist learning theories [20].

In this article, we apply the ideas of the extended cognition hypothesis to the use of computational thinking paradigm in computer science education. For a more complete review of extended mind hypothesis and its applications we refer to textbooks by Mark Rowland [16] and Robert Rupert [21].

2 Analysis

The common denominator for both the computational thinking and the extended cognition is technology, especially the relationship and interaction between human cognition and technology.

From the extended cognition point of view, technology is an integral part of the human cognition. It was built and developed to help humans function in the environment. The human mind actively utilizes these available external features that can be used as cognitive tools and integrates them into cognitive processes.

Quite opposite to the extended cognition point of view, computational thinking approach sees a very strong distinction between the human and the computer. Computers use a different way of logic and technology, something that is not natural for humans. Computational thinking is seen as a tool or a skill to be used to understand and utilize this external technology.

This can be seen for example in the original article by Wing, where she emphasizes that computational thinking is "A way that humans, not computers, think" [6]. The statement can be considered to be actual, at least at the current level of technology, but it still underlines the difference between the human and the machine. This is something that contradicts the external cognition point of view.

To understand better this dichotomy, let us consider some special cases of human-computer relationship.

2.1 Extended limits of computational thinking

From the everyday classroom perspective the adaption of computers may seem like a rapid process. Computational thinking is, therefore, because of this view and for practical reasons, understood and taught using current technology.

But computational systems have had a long history of technological development. Computational systems have existed long before the arrival of digital computers and the development and use of digital computers have seen several extensive revolutions. We may expect that the landscape will not stay stationary in the near future either.

For the extended cognition hypothesis, there is a natural way to expand it to different levels of technology. For example, to remember a phone number:

Without any tools. One has to remember the number using cognitive functions of the brain. This requires the conscious use of chunking techniques to make the number memorable [22]. It also requires recalling the number from memory every now and then to prevent forgetting [23]. In effect, it requires concentration and can result in failure if the person is exposed to cognitive stress or distractions.

A pen and paper. One can write the number down. This eliminates most of the cognitive load caused by the task. The skill required for the task is the ability to write numbers and understand written numbers.

Modern smartphones. The phone number with required contact details are transmitted automatically. Cognitive requirements depend on the usability of the smartphone, but can theoretically be set arbitrarily low.

Future implant technology Speculative future technology of brain implants is often used as a thought experiment to show ultimate possibilities of how extended cognition might work in the future [15]. A proactive technology anticipates the required action. This could optionally be external agents such as robots or artificial intelligence that serve human needs.

In addition to memory, another example of extended cognition is performing calculations where a pen and a paper, a pocket calculator, or calculator implanted in the human body is used to extend human cognitive capacity.

For computational thinking, such thought experiments using different levels of technology are less often presented. Even though using algorithms to perform mathematical calculations was already used 4000 years ago [24].

Let us consider some tasks where CSTA definition features of computational thinking presented in chapter 1.1 are used.

Building a house. Building a house requires creating models and organizing plans. Even when human labor is used, repetitive tasks are executed in a rather similar way to which programmable machines work in modern automated factories. Building process also involves problem-solving with technical constraints.

Building a business. Business organizations need multi-level hierarchical organization. Standardized practices and hierarchy create fault tolerance to the system. A highly detailed theory of organizations has been created to help to design and to implement the creation and development of business organizations [25]. All the features of computational thinking are clearly used in governing business organizations.

Building software. Software engineering is the practical side of computer science, and clearly has an affect on how we view computational thinking. But in the same way that we develop analogies to how other engineering and business systems resemble computers, we might also understand computer systems similar to engineering and businesses. Computers just provide us with cheap, tireless workers who need very specific instructions to work.

Building a society. Governing a society and creating an efficient administration for it is a task requiring the solving of many economical and managerial

problems. Building, for example, a taxation system or public services requires skills in logic, engineering, and system modeling while the work is also related to social issues and ethics. As just described, this is a non-technical task with high requirements in computational thinking skills as defined by CSTA.

One more good example of human activity closely related to computational thinking without digital computers is many recreational games human play. Rules are created to make a game fair and interesting. The game is executed using humans as the operating agents until the game reaches an objective such as a win or a tie.

Together these two dimensions illuminate the field where human activity is related to computational thinking skills and where the ideas of extended cognition must be applied respectively.

2.2 Levels of computer programming

Using the computational thinking term definition by Wing presented in section 1.1, the core idea of the computational thinking is to formulate problems so that they can be solved by information-processing agents such as digital computers. The reasoning is that digital computers can be used to solve problems in many fields and in everyday tasks.

Computer programming is often not considered to be a computational thinking skill by itself. But it is a closely related and often indistinguishable from other computational thinking skills. Let us, therefore, review some computer programming languages and paradigms.

Computer programs were originally hand-coded in binary machine language. It is easy to see how this differs from modern computer programming, with automatic code completion, syntax highlighting, and other convenient features.

More drastic changes have taken place in the application domains. While early computers were isolated machines with little or no user interface, intended for mathematical calculations, modern computers are connected to the Internet, have modern graphical user interfaces and IO devices, and can be used for almost any purpose from computer games and multimedia applications to running network services and social networks.

Let us consider some examples how computer programming has changed from the viewpoint of extended cognition hypothesis.

Low-level language. Primitive languages such as assembler language process information at a very low level. The programmer needs to keep track of the program states in their mind and emulate the execution process.

High-level language. High-level languages hide much of the low-level details in programming and let the programmer concentrate on the programming logic. This allows the building of more complex program for the same cognitive load.

Integrated development environment. When programming is done with a plain text editor, it is the responsibility of the programmer to keep the syntax of the code error free. The programmer also needs to remember or look up the names of variables, functions, etc. Modern integrated development environment (IDE) takes care of these tasks and provide such features as intelligent code completion and help functions to speed up the development process. This frees up cognitive resources for novice programmers and programmers in new coding environments.

Code libraries and frameworks. Modern computers and computing environments including operating systems are highly complex systems. Ready-made code libraries and frameworks are used to create maintainable and scalable programs. This allows the programmer to do work on the higher level of code abstraction.

Internet and social networks. Before the Internet and advanced web services with large user bases, computer programmers were isolated from the programming communities. A programmer needed to read manuals or other literature to learn more about programming topics. Nowadays Google, StackExchange, and other web services provide huge amounts of information. There are ready-made code examples from almost every subject. For the programmer, this allows tools for using alternative problem-solving methods in programming tasks. New skills are needed such as managing and searching large amounts of information and social co-operation skills.

These changes are not independent but they act together. The increasing range of programming paradigms help with problems such as code complexity and re-usability, concurrent programming, and asynchronous code execution. For example, functional programming hides the program execution flow from the user allowing higher level abstraction requiring the use of very different set of cognitive processes.

In the future automatic program generation and artificial intelligence will move computer programming from actual coding to developing technical requirements for the program. Together with artificial intelligence, this will probably move programming away from logic and mathematics, and closer to being a linguistic endeavor.

Another near-future trend in programming is proof systems. The correctness of computer programs can be formally verified using formal methods of mathematics [26]. Using proof systems removes much of the technical details of algorithms from programming, allowing for a higher level of abstraction.

2.3 Modes of technology

Technology can operate on several fundamentally different modes such as textual, visual, and auditory. In further examination, for example, temporal and metadata level aspects can change the modality of the technology. Let us consider some examples.

Name of a bird species. Naming a bird species is naturally expressed as a text. Symbols needed to present the name can be written down. Either numerical or alphabetical symbols can be used depending on the application.

The visual appearance of a bird. It is possible to sketch the bird on a paper. Even if an exact reproduction is impossible, most prominent features can be drawn so that the image will help in the identification of the bird later.

The sound of the bird. The sound of the bird can be described in words but a direct presentation is difficult without a phonetic script suitable for the task.

A database of features for the identification of birds. Here the second level of abstraction is needed to organize the data meaningfully. Namely, structured metadata is needed to bind together the different modes of features.

From the viewpoint of extended cognition, these examples illustrate how human cognition can be naturally extended by technology in all the modalities we already utilize. The use of technology is not isolated into a certain modality of human existence.

Computational thinking, on the other hand, is very indifferent regarding the modalities. In the way computational thinking is taught, using algorithms and programming, it ignores much of the knowledge of the structure of the world. Dealing with modalities is left to the application level tasks.

Keeping computational thinking insensitive to the modes of technology may seem like a rational choice. This retains the tool used distinct from the subject operated. But according to the extended cognition principle, there is no distinction between what is inside the cognition and what is outside the cognition. This dividing way of thinking separates the human cognition from the technology and undermines their potential efficient cooperation.

From the extended cognition hypothesis we can thus conclude that understanding and internalizing the human-computer interaction using the whole scale of modalities is essential to the efficiency of extended cognition.

One research example of efficiency gained by practicing modalities is the connection between spatial thinking and using 3D computer games. In earlier studies, it has been shown that women do not perform as well as men on some spatial tasks such as mental rotation [27]. Spence et al. showed that fundamentally the learning rate of women is not less than that of men [28]. The difference in skills come from cultural aspects. This has further consequences as spatial thinking is related in learning skills in mathematics and science [29]. If computational thinking in education stresses too much programming logic and understanding algorithms and assumes that data is already in computer processable format, much of the possible interaction is missed.

To improve the situation there are several options. Human-computer interaction (HCI) is a well-established field of science. It is a discipline working together with computer science, cognitive science, psychology. Calderon et al. present a

methodology to systematically introduce this topic to students of early age to complement the ideas of computational thinking [30].

In addition to formal HCI methods, human perception should be approached in a systematic way discussing underlying principles of physics, how the human nervous system is stimulated, and how the mind perceives and interprets the sensory information. This provides a more concise view of multimodal interaction with technology.

3 Discussion

Regarding computer science education the main question becomes how to enable efficient use of technology to support cognitive processes in the adoption of skills? What are the technological features that allow us to perform better in these tasks? And how the ideas of extended cognition help us to answer these questions.

Let us revisit the computational thinking definition by CSTA from section 1.1 and consider the interaction of human cognition and technology.

The first claim of the definition says that computational thinking is *"formulating problems in a way that enables us to use a computer and other tools to help solve them"*. From an extended cognition viewpoint, this could be expressed more generally that in education we want students to use technology and solve problems. No special thinking should be required for using technology.

For example, using a calculator to do calculations from very early age gives a different type of thinking that is already adapted to formulating problems adequately. For a person who doesn't have a calculator, large multiplications and divisions are hard problems. With a calculator these calculations are trivial. Doing a large number of calculations is still a hard task even using a calculator but having numbers available digitally and using a spreadsheet program makes this task trivial.

The first claim is closely related to the third claim which states that computational thinking is *"representing data through abstractions such as models and simulations"*. As we saw in section 2.1, computational thinking is not related to digital computers generally. We just use the term often in the context of computer science. Humans already use models and simulations in many activities, such as social relationships, games, economic planning, and in creative tasks, even as simple as cooking or expressing oneself through art.

In many cases, abstract thinking arises spontaneously from using technology. For example, social relationships can be visualized using social media applications such as Facebook [31]. Transactions of a bank account can often be downloaded from an online bank in CSV format. Our mobile phones can track the GPS data of our daily movement.

The deeper understanding of data models is something that does not always arise by itself. But often technology already provides tools to utilize the data. For example different parameters can be calculated and visualizations can be made from GPS data without a deeper understanding. Using the data teaches

the properties it possesses. Requirements of use cases define the data models needed for the data. From this point of view, the extended cognition view of thinking encourages declarative and functional thinking versus traditional procedural thinking.

The second claim, one we left without addressing earlier, is that computational thinking includes *"logically organizing and analyzing data"*. From the extended cognition point of view, it is not as necessary to emphasize the logical organizing of the data from the viewpoint that sees human and technology as separated entities. In the extended cognition view, the organization of the data is largely a natural consequence of using technology.

The fourth claim of the definition is that computational thinking is *"automating solutions through algorithmic thinking (a series of ordered steps)"*. Algorithmic thinking as a series of ordered steps expresses the close connection of the definition to the history of computer science and software engineering. Most programming, most notable programming with low-level languages, is procedural programming consisting of series of ordered steps or commands.

While procedural programming is the de facto practical programming paradigm, it should be noted that it is not the only one and might not be cognitively the most natural paradigm. Functional programming is a paradigm that more naturally arises from the use of data and models. There are also automatic code generation and proof systems that allow solving computational tasks without the low-level understanding of programming as mentioned in section 2.2. It is therefore not clear that using technology promotes procedural thinking as expressed in this fourth claim of the definition.

The fifth claim of the definition is that computational thinking is *"identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources"*. The task of optimizing computational efficiency is one of the key differences between mathematics and computer science according to Knuth [4]. It could be argued that while this is one of the key concepts of computer science, from the extended cognition hypothesis point of view, it is more of an auxiliary problem for most of the application domains.

The sixth and final claim of the definition is there should be *"generalizing and transferring this problem-solving process to a wide variety of problems"*. From the extended cognition view of computation, there is an intrinsic means of computational optimization and generalization in that humans can choose from the technology they use. Human cognition is not limited to its natural capacity but there is a drive to find utilize optimal external resources. This is not to dismiss the fifth and sixth claim of the definition but to offer an insight of what is already implied in the adoption of advanced use of technology. The extended cognition view provides a higher level of abstraction in that it is not about algorithmic efficiency but choosing right technological tools and adopting them accordingly to the problem-solving processes.

4 Conclusions

There are many implicit and explicit attributes in the use of the term computational thinking that express the historical constraints of the term. The term comes from software engineering traits and has strong notes of separate human, the programmer or the user, and the computer. Humans need skills to use computers, which are external machines to help humans with their tasks.

This highly contradicts the view of extended cognition where technology is an integral part of the extended human. An example of technological embodiment is the collision of cars. A car is a functional extension of our body and if we are involved in the collision we probably describe the situation as "another car hitting me", and not "another car hitting the car I am in". The same embodiment happens notably in computer games, virtual reality applications, and many other uses of advanced technology.

To understand an efficient computer science education we need deep understanding of what is the relationship between students and technology. We must ask how thinking has changed and how it can be changed by the use of technology. This is the pedagogical content knowledge of 21st-century skills [32].

For this, we suggest the concept *technology synchronized thinking* to complement the idea of computational thinking and to reflect the idea of extended cognition hypothesis that external technology can be seen as part of human cognitive functions.

The idea is that to promote the possibilities of functional extended cognition, a cooperation or synchronization must be developed between the cognitive processes of the mind and the technology. This is achieved using cognitively suitable technologies. It also requires using problem-solving and engineering tasks that adapt the brain for the technological environment.

1. Computation. Traditionally computer programming has been considered something that is created in the human mind and projected onto external computational machines. For example, Alan J. Perlis writes *"Every computer program is a model, hatched in the mind, of a real or mental process. These processes, arising from human experience and thought, are huge in number, intricate in detail, and at any time only partially understood. They are modeled to our permanent satisfaction rarely by our computer programs."* [33]. This one-way interaction is historically related to the low level and compiled programming languages. Instead, programming languages with interactive interpreters are more suitable for creating productive extended cognitive functions. Functional programming paradigm provides a more cognitively compatible way of organizing functions and allows removing strict step-by-step of procedural programming. The compatibility is based on the associative quality of human memory. Much of the low-level information processing in the human mind is done automatically and unconsciously.
2. Models. Almost all the information we use today is digital. This allows easy data processing using computers. Most computer science education approaches this data processing from low-level details such as data formats

and low-level algorithms. From the extended cognition point of view higher abstraction level, top-down approaches are preferred. This means that data is processed using practical applications and use cases. Low-level details are not considered at early stages. While details are interesting from many computational aspects, they do not support well the cognitive entanglement of external technology and human cognitive processes.

3. HCI. Traditionally computational thinking and programming are considered as textually expressed symbolic processes. While symbolic presentation allows precise syntax, it doesn't fully utilize human cognitive capacity. As discussed in section 2.3, multimodal and 3D human-computer interfaces provide more integrated experience. This experience allows higher level abstraction in human cognitive processes increasing its capabilities as seen in the example concerning the connection between 3D games and mathematical thinking.

In the teaching of computational thinking, there are many methods that do not use computers or technology but games or other tasks related to the ideas of computational thinking [34, 35]. These methods are sometimes called computational thinking unplugged [36, 37].

From the extended cognition point of view, it is not advisable to separate computational thinking ideas from technology. This view has some experimental evidence. For example Grover et al. write about computational thinking tools not utilizing computers: *"Noteworthy efforts like CS Unplugged that introduce computing concepts without the use of a computer, while providing valuable introductory activities for exposing children to the nature of CS, may be keeping learners from the crucial computational experiences involved in CTs common practice."* [38]

Peter J. Denning writes also about the value of computer science: *"We are most valued not for our computational thinking, but for our computational doing."* [39] These views reassert the idea of extended cognition hypothesis that skills needed in using modern digital technology are not separable from the technology itself.

Another aspect related to the extended cognition hypothesis is the socially extended mind hypothesis which extends the cognitive realm to the social domain. The socially extended mind builds on the enactive idea of social affordances. The idea is very strong since human behavior depends greatly on the cultural and the social context. It is impossible to study human behavior meaningfully without considering social and cultural bonds and prospects.

The socially extended mind is an especially current topic as the Internet and social media has changed the way how people interact with each other globally. There is also more intelligent technology for people to interact with, such as commercial online chatbots and personal assistant technology like Apple Siri, Google Assistant, Amazon Alexa, and Microsoft Cortana.

More research is needed on how human social networks and cognitive technology transforms the concept of computational thinking. The phenomenon is bidirectional. The cultural change gives a feedback and guides the technological development. This can lead to subcultures which differ both in technology and

in user culture. This makes a single universal skill-set impossible to define in some cases.

References

1. Matti Tedre and Peter J Denning. Shifting identities in computing: From a useful tool to a new method and theory of science. In *Informatics in the Future*, pages 1–16. Springer, 2017.
2. E. W. Dijkstra. Programming as a discipline of mathematical nature. *The American Mathematical Monthly*, 81(6):608–612, 1974.
3. Donald E Knuth. Computer science and its relation to mathematics. *The American Mathematical Monthly*, 81(4):323–343, 1974.
4. Donald E Knuth. Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly*, 92(3):170–181, 1985.
5. Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.
6. Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.
7. Jeannette M. Wing. Research notebook: Computational thinking–what and why? — carnegie mellon school of computer science. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why, accessed: 2018-01-14. Accessed: 2018-01-14.
8. Alfred V Aho. Computation and computational thinking. *The Computer Journal*, 55(7):832–835, 2012.
9. Chris Stephenson and Valerie Barr. Defining computational thinking for k12. *Special Issue – Computer Science K8: Building a Strong Foundation*, 2012.
10. Matti Tedre and Peter J Denning. The long quest for computational thinking. In *Koli Calling*, pages 120–129, 2016.
11. John Palmer. Parmenides. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 edition, 2016.
12. Bryan Magee. *The great philosophers: An introduction to western philosophy*. Oxford University Press on Demand, 2000.
13. Michael A Paradiso, Mark F Bear, and Barry W Connors. *Neuroscience: exploring the brain*. Philadelphia : Wolters Kluwe, 2016.
14. Fred Dretske. Phenomenal externalism or if meanings ain't in the head, where are qualia? *Philosophical Issues*, 7:143–158, 1996.
15. Andy Clark and David Chalmers. The extended mind. *analysis*, 58(1):7–19, 1998.
16. Mark Rowlands. The new science of the mind: From extended mind to embodied phenomenology (bradford books). 2016.
17. Eric Arnau, Anna Estany, Rafael González del Solar, and Thomas Sturm. The extended cognition thesis: Its significance for the philosophy of (cognitive) science. *Philosophical Psychology*, 27(1):1–18, 2014.
18. Robert A Wilson. Ten questions concerning extended cognition. *Philosophical psychology*, 27(1):19–33, 2014.
19. Joseph Mikhael. *Philosophy of Bioinformatics: Extended Cognition, Analogies and Mechanisms*. ProQuest, 2007.
20. Douglas L Holton. Constructivism+ embodied cognition= enactivism: theoretical and practical implications for conceptual change. In *AERA 2010 Conference*, 2010.

21. Robert D Rupert. *Cognitive systems and the extended mind*. Oxford University Press, 2009.
22. George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
23. Erik M Altmann and Wayne D Gray. Forgetting to remember: The functional relationship of decay and interference. *Psychological science*, 13(1):27–33, 2002.
24. Donald E Knuth. Ancient babylonian algorithms. *Communications of the ACM*, 15(7):671–677, 1972.
25. Richard L Daft. *Organization theory and design*. Cengage learning, 2015.
26. Adam Chlipala, Benjamin Delaware, Samuel Duchovni, Jason Gross, Clément Pit-Claudel, Sorawit Suriyakarn, Peng Wang, and Katherine Ye. The end of history? using a proof assistant to replace language design with library design. In *SNAPL'17: Proceedings of the The 2nd Summit oN Advances in Programming Languages*, May 2017.
27. Daniel Voyer, Susan Voyer, and M Philip Bryden. Magnitude of sex differences in spatial abilities: a meta-analysis and consideration of critical variables., 1995.
28. Ian Spence, Jingjie Jessica Yu, Jing Feng, and Jeff Marshman. Women match men when learning a spatial skill. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 35(4):1097, 2009.
29. Nora S Newcombe. Picture this: Increasing math and science learning by improving spatial thinking. *American Educator*, 34(2):29, 2010.
30. Ana C Calderon and Tom Crick. Using interface design to develop computational thinking skills. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, pages 127–129. ACM, 2015.
31. Stephen Wolfram. Personal analytics for facebook. http://blog.wolframalpha.com/2012/08/30/wolframalpha-personal-analytics-for-facebook/, 2012. Accessed: 2018-01-13.
32. Lee S Shulman. Those who understand: Knowledge growth in teaching. *Educational researcher*, 15(2):4–14, 1986.
33. Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and interpretation of computer programs*. Justin Kelly, 1996.
34. Timothy C Bell, Ian H Witten, and Mike Fellows. *Computer Science Unplugged: Off-line activities and games for all ages*. Computer Science Unplugged, 1998.
35. Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, 13(1):20–29, 2009.
36. Yvon Feaster, Luke Segars, Sally K Wahba, and Jason O Hallstrom. Teaching cs unplugged in the high school (with limited success). In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 248–252. ACM, 2011.
37. Paul Curzon, Peter W McOwan, Nicola Plant, and Laura R Meagher. Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th workshop in primary and secondary computing education*, pages 89–92. ACM, 2014.
38. Shuchi Grover and Roy Pea. Computational thinking in k–12: A review of the state of the field. *Educational Researcher*, 42(1):38–43, 2013.
39. Peter J Denning. The profession of it beyond computational thinking. *Communications of the ACM*, 52(6):28–30, 2009.